SYSTEMS AND METHODS FOR NETWORK ADDRESS FAILOVER

Related Files

This invention is related to application serial number 10/128,656, filed April 22, 2002, entitled "SCSI-BASED STORAGE AREA NETWORK" (Attorney Docket No.: 1370.021US1), application serial number 10/131,793, filed April 22, 2002, entitled "VIRTUAL SCSI BUS FOR SCSI-BASED STORAGE AREA NETWORK" (Attorney Docket No.: 1370.022US1), provisional application serial number 60/374,921, filed April 22, 2002, entitled "INTERNET PROTOCOL CONNECTED STORAGE AREA NETWORK" (Attorney Docket No.: 1370.028PRV), application serial number 10/356,073, filed January 31, 2003, entitled "STORAGE ROUTER WITH INTEGRATED SCSI SWITCH" (Attorney Docket No.: 1370.039US1), and application serial number 10/128657, filed Apr 22, 2002, entitled "METHOD AND APPARATUS FOR EXCHANGING CONFIGURATION INFORMATION BETWEEN NODES OPERATING IN A MASTER-SLAVE CONFIGURATION" (Attorney Docket No.: 1370.027US1) all of the above of which are hereby incorporated by reference.

Field

This invention relates generally to network addressing, and more particularly to providing address failover capability for network interfaces on an application gateway device.

Background

Many devices capable of being attached to a network such as personal computers, servers, routers and switches have more than one network interface. Typically multiple network interfaces may be used by the network device to provide connectivity to differing networks or systems, to provide a redundant path to a network, or they may be used to provided increased network throughput (i.e. increased bandwidth).

Occasionally a network interface may fail. When this happens, software applications using the network interface are no longer able to use the network interface to send and receive data, possibly resulting in the failure of the software application.

In some systems, when a network interface fails, the system attempts to migrate the software application to another network device on the same network as the device experiencing the network interface failure. The application then runs on the new network device, often in a manner that is transparent to the users on the system. The ability to migrate an application to a new device is sometimes referred to as "failover."

Failover capability is useful in providing fault tolerant applications, however there are problems associated with failing over to a second network device. Often it takes a substantial amount of time to accomplish the failover, because application configuration and data must be transferred to the second network device. A user will often notice a delay in the response of the system while the failover takes place. In addition, network connections between the failed over application and other hosts and applications may need to be reestablished because the new application will reside on a network device having a different network address than the original network device. This also can take a substantial mount of time and may result in the loss of data.

In view of the above problems and issues, there is a need in the art for the present invention.

Summary

The above-mentioned shortcomings, disadvantages and problems are addressed by the present invention, which will be understood by reading and studying the following specification.

Systems and methods provide network address failover capability within an application gateway device. In one aspect, a system has a first network interface and a second network interface. The system receives a set of configuration data, the configuration data may include a first network address for the first network interface and a second network address for the second network interface. At startup or during later operation, the system may detect the failure of the first network interface. The configuration data may be analyzed to determine if the first network address can be used on the second network interface. If so, the first network address is moved from the first network interface to the second network interface.

The present invention describes systems, methods, and computer-readable media of varying scope. In addition to the aspects and advantages of the present invention described in this summary, further aspects and advantages of the invention will become apparent by reference to the drawings and by reading the detailed description that follows.

Brief Description Of The Drawings

FIG. 1A is a block diagram of a hardware and operating environment for a storage router application gateway device in which different embodiments of the invention can be practiced.

FIG. 1B is a block diagram of a clustered storage router hardware and operating environment in which different embodiments of the invention can be practiced.

FIG. 2 is a block diagram of the major hardware components of a storage router according to an embodiment of the invention.

FIG. 3A is a flowchart illustrating a method for failing over a network address according to an embodiment of the invention.

FIG. 3B is a flowchart providing further details on a method moving a network address from a first network interface to a second network interface according to an embodiment of the invention.

Detailed Description

In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical and other changes may be made without departing from the scope of the present invention.

Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, terms such as “processing” or “computing” or “calculating” or “determining” or “displaying” or the like, refer to the action and processes of a computer system, or similar computing device, that manipulates and transforms data represented as

physical (e.g., electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

In the Figures, the same reference number is used throughout to refer to an identical component which appears in multiple Figures. Signals and connections may be referred to by the same reference number or label, and the actual meaning will be clear from its use in the context of the description.

The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

Operating Environment

Some embodiments of the invention operate in an environment of systems and methods that provide a means for Fibre Channel based Storage Area Networks (SANs) to be accessed from TCP/IP network hosts. FIG. 1A is a block diagram describing the major components of such a system. In these embodiments, storage router system 100 includes computers (127, 128) connected through an IP network 129 to one or more instances of storage router 110. Storage router 110 comprises an application gateway device that is connected in turn through storage network 130 to one or more SCSI devices 140. In some embodiments, each instances of a storage router 110 may have its own TCP/IP network address. For the purpose of this specification, an application gateway device comprises a device that receives data conforming to a first protocol and processes the data to conform to a second differing protocol. In the embodiment shown in Fig. 1A, storage router 110 includes an iSCSI interface 104, a SCSI router 105 and a SCSI interface 106. iSCSI interface 104 receives encapsulated SCSI packets from IP network 129, extracts the SCSI packet and send the SCSI packet to SCSI router 105. SCSI interface 106 modifies the SCSI packet to conform to its network protocol (e.g., Fibre Channel, parallel SCSI, or iSCSI) and places the modified SCSI packet onto storage network 130. The SCSI packet is then delivered to its designated SCSI device 140.

In one embodiment, storage router 110 provides IPv4 router functionality between a Gigabit Ethernet and a Fibre Channel interface. In one such embodiment, static routes are supported. In addition, storage router 110 supports a configurable MTU size for each interface, and has the ability to reassemble and refragment IP packets based on the MTU of the destination interface.

In one embodiment, storage router 110 acts as a gateway, converting SCSI protocol between Fibre Channel and TCP/IP. Storage router 110 is configured in such an embodiment to present Fibre Channel devices as iSCSI targets, providing the ability for clients on the IP network to directly access storage devices.

In one embodiment, SCSI routing occurs in the Storage Router 110 through the mapping of physical storage devices to iSCSI targets. An iSCSI target (also called logical target) is an arbitrary name for a group of physical storage devices. Mappings between an iSCSI target to multiple physical devices can be established using configuration programs on storage router 110. An iSCSI target always contains at least one Logical Unit Number (LUN). Each LUN on an iSCSI target is mapped to a single LUN on a physical storage target.

In operation, if a network interface on storage router 110 fails, the SCSI router instances using the interface may have their respective IP network addresses failed over to a secondary network interface. For example, assume that the network interface being used by SCSI router 105.2 fails. The IP network address associated with SCSI router 105.2 may be moved (i.e. failed over) to the same network interface as SCSI router 105.1. The movement is generally transparent both to the SCSI router instance 105.2, and to hosts and applications that are communicating via the network to SCSI router instance 105.2. Further details on the failover of the IP network address are provided with reference to FIGs. 3A and 3B below.

FIG. 1B is a block diagram of a clustered storage router hardware and operating environment in which different embodiments of the invention can be practiced. In some embodiments of the invention, high-availability network interface 111 interconnects the storage routers 110 that participate as members in a cluster. In some embodiments, the high-availability network interface 111 is an Ethernet interface, however the invention is not limited to any particular network type. In some embodiments, members of cluster 112 share

configuration information and heartbeat information through high availability interface 111. This configuration information may include IP network addresses for each SCSI router instance 105 that is to operate on each storage router 110. In addition, the configuration information may include a primary and secondary network interface for each SCSI router instance.

Although the exemplary environment illustrates two members 110.1 and 110.2 of cluster 112, the invention is not limited to any particular number of members of a cluster.

Further details on the operation of the above can be found in United States Patent Application serial number 10/131,793 entitled "VIRTUAL SCSI BUS FOR SCSI-BASED STORAGE AREA NETWORK" (Attorney Docket No.: 1370.022US1) and in United States Patent Application serial number 10/356,073 entitled "INTEGRATED STORAGE ROUTER AND FIBRE CHANNEL SWITCH" (Attorney Docket No.: 1370.039US1), both of which have been previously incorporated by reference.

Figure 2 is a block diagram providing further details of the major hardware components comprising storage router 110. In some embodiments, a storage router 110 includes a router portion 210 and a switch portion 220 on a common motherboard 200. The motherboard is powered by a power supply (not shown) and cooled by common cooling system, such as a fan (also not shown).

Router portion 210, which in the exemplary embodiment complies with draft 08 and later versions of the iSCSI protocol and incorporates commercially available router technology, such as the 5420 and 5428 Storage Routers from Cisco Systems, Inc. of San Jose, California, includes Gigabit Ethernet (GE) ports 211.1 and 211.2, console port 212, management port 213, high-availability (HA) port 214, bridge-and-buffer module 215, interface software 216, router processor 217, and router-to-switch interface 218.

GE ports 211.1 and 211.2 couple the storage router to an IP network for access by one or more servers or other computers, such as servers or iSCSI hosts (in Figure 1). In some embodiments, GE ports 211.1 and 211.2 have respective MAC addresses, which are determined according to a base MAC address for the storage router plus 31 minus the respective port number. Two or more Gigabit Ethernet interfaces may be available. In some

embodiments, one or more of the Gigabit Ethernet interfaces may provide internal support for maintaining Virtual Local Area Networks (VLANs). Each SCSI router typically supports a single IP address. The SCSI router IP address may be tied to any network (or VLAN) on either GE interface. Generally at least one SCSI router instance is created for each GE interface.

Console port 212 couples to a local control console (not shown). In the exemplary embodiment, this port takes the form of an RS-232 interface.

Management port 213 provides a connection for managing and/or configuring storage router 110. In the exemplary embodiment, this port takes the form of a 10/100 Ethernet port and may be assigned the base MAC address for the router-switch.

HA port 214 provides a physical connection for high-availability communication with another router-switch, such as storage router 110 in Figure 1. In the exemplary embodiment, this port takes the form of a 10/100 Ethernet port, and is assigned the base MAC address plus 1.

Bridge-and-buffer module 215, which is coupled to GE ports 211.1 and 211.2, provides router services that are compliant with draft 08 and later versions of the iSCSI protocol. In the exemplary embodiment, module 215 incorporates a Peripheral Component Interface (PCI) bridge, such as the GT64260 from Marvell Technology Group, LTD. of Sunnyvale, California. Also module 215 includes a 64-megabyte flash file system, a 1-megabyte boot flash, and a 256-megabyte non-volatile FLASH memory (not shown separately.) Configuration memory 230 may be part of the flash file system, the boot flash or the non-volatile flash memory, or it may be a separate non-volatile flash memory. In addition, in alternative embodiments, configuration memory 230 may be part of a hard disk, CD-ROM, DVD-ROM or other persistent memory (not shown). The invention is not limited to any particular type of memory for configuration memory 230.

In addition to data and other software used for conventional router operations, module 215 includes router-switch interface software 216. Router-switch software 216 performs iSCSI routing between servers and the storage devices. The software includes an integrated router-switch command line interface module CLI and a web-based graphical-user-interface

module (GUI) for operation, configuration and administration, maintenance, and support of the router-switch 110. Both the command-line interface and the graphical user interface are accessible from a terminal via one or both of the ports 213 and 214. Additionally, to facilitate management activities, interface software 216 includes an SNMP router-management agent AGT and an MIB router handler HD. (SNMP denotes the Simple Network Management Protocol, and MIB denotes Management Information Base (MIB)). The agent and handler cooperate with counterparts in switch portion 220 (as detailed below) to provide integrated management and control of router and switching functions in router-switch 200.

Router Processor 217, in the exemplary embodiment, is implemented as a 533-MHz MPC7410 PowerPC from Motorola, Inc. of Schaumburg, Illinois. This processor includes 1-megabyte local L2 cache (not shown separately). In the exemplary embodiment, router processor 217 runs a version of the VX Works operating system from WindRiver Systems, Inc. of Alameda, California. To support this operating system, the exemplary embodiment also provides means for isolating file allocations tables from other high-use memory areas(such as areas where log and configuration files are written).

Coupled to router processor 217 as well as to bridge-and-buffer module 215 is router-to-switch (RTS) interface 218. RTS interface 218 includes N/NL switch-interface ports 218.1 and 218.2 and management-interface port 218.3, where the port type of N or NL is determined by negotiation. N type ports may act as a Fibre Channel point to point port, NL type ports may negotiate as a loop.

Switch-interface ports 218.1 and 218.2 are internal Fibre Channel (FC) interfaces through which the router portion conducts I/O operations with the switch portion. When a mapping to a FC storage device is created, the router-switch software automatically selects one of the switch-interface ports to use when accessing the target device. The internal interfaces are selected at random and evenly on a per-LUN (logical unit number) basis, allowing the router-switch to load-balance between the two FC interfaces. The operational status of these internal FC interfaces is monitored by each active SCSI Router application running on the switch-router. The failure of either of these two interfaces is considered a unit failure, and if the switch-router is part of a cluster, all active SCSI Router applications will fail

over to another switch-router in the cluster. Other embodiments allow operations to continue with the remaining switch-interface port. Still other embodiments include more than two switch-interface ports.

In the exemplary embodiment, the N/NL switch-interface ports can each use up to 32 World Wide Port Names (WWPNs). The WWPNs for port 218.1 are computed as 28 + virtual port + base MAC address, and the WWPNs for port 218.2 are computed as 29 + virtual port + base MAC address. Additionally, switch-interface ports 218.1 and 218.2 are hidden from the user. One exception is the WWPN of each internal port. The internal WWPNs are called "initiator" WWPNs. Users who set up access control by WWPN on their FC devices set up the device to allow access to both initiator WWPNs.

Switch-interface port 218.3 is used to exchange configuration data and get operational information from switch portion 220 through its management-interface port 224. In the exemplary embodiment, switch-interface port 218.3 is an 10/100 Ethernet port. In the exemplary embodiment, this exchange occurs under the control of a Switch Management Language (SML) Application Program Interface (API) that is part of interface software 216. One example of a suitable API is available from QLogic Corporation of Aliso Viejo, California. Ports 218.1, 218.2, and 218.3 are coupled respectively to FC interface ports 221.1 and 221.2 and interface port 224 of switch portion 220.

Switch portion 220, which in the exemplary embodiment incorporates commercially available technology and supports multiple protocols including IP and SCSI, additionally includes internal FC interface ports 221.1 and 221.2, an FC switch 222, external FC ports (or interfaces) 223.1-223.8, a management interface port 224, and a switch processor module 225.

FC interface ports 221.1 221.2 are coupled respectively to ports of 218.1 and 218.2 of the router-to-switch interface via internal optical fiber links, thereby forming internal FC links. In the exemplary embodiment, each FC interface supports auto-negotiation as either an F or FL port.

FC switch 222, in the exemplary embodiment, incorporates a SANbox2-16 FC switch from QLogic Corporation. This SANbox2 switch includes QLogic's Itasca switch ASIC

(application-specific integrated circuit.) Among other things, this switch supports Extended Link Service (ELS) frames that contain manufacturer information.

FC ports 223.1-223.8, which adhere to one or more FC standards or other desirable communications protocols, can be connected as point-to-point links, in a loop or to a switch. For flow control, the exemplary embodiment implements a Fibre Channel standard that uses a look-ahead, sliding-window scheme, which provides a guaranteed delivery capability. In this scheme, the ports output data in frames that are limited to 2148 bytes in length, with each frame having a header and a checksum. A set of related frames for one operation is called a sequence.

Moreover, the FC ports are auto-discovering and self-configuring and provide 2-Gbps full-duplex, auto-detection for compatibility with 1-Gbps devices. For each external FC port, the exemplary embodiment also supports: Arbitrated Loop (AL) Fairness; Interface enable/disable; Linkspeed settable to 1 Gbps, 2 Gbps, or Auto; Multi-Frame Sequence bundling; Private (Translated) Loop mode.

Switch processor module 225 operates the FC switch and includes a switch processor (or controller) 225.1, and associated memory that includes a switch management agent 225.2, and a switch MIB handler 225.3. In the exemplary embodiment, switch processor 225.1 includes an Intel Pentium processor and a Linux operating system. Additionally, processor 225 has its own software image, initialization process, configuration commands, command-line interface, and graphical user interface (not shown). (In the exemplary embodiment, this command-line interface and graphical-user interface are not exposed to the end user.) A copy of the switch software image for the switch portion is maintained as a tar file 226 in bridge-and-buffer module 215 of router portion 210.

Further details on the operation of the above describe system, including high availability embodiments can be found in application serial number 10/128,656, entitled "SCSI-BASED STORAGE AREA NETWORK" (Attorney Docket No.: 1370.021US1), application serial number 10/131,793, entitled "VIRTUAL SCSI BUS FOR SCSI-BASED STORAGE AREA NETWORK" (Attorney Docket No.: 1370.022US1), and provisional application serial number 60/374,921, entitled "INTERNET PROTOCOL CONNECTED

STORAGE AREA NETWORK" (Attorney Docket No.: 1370.028PRV), all of which have been previously incorporated by reference.

FIGs. 3A and 3B are a flowcharts illustrating methods according to embodiments of the invention for providing network address failover capability. The methods to be performed by the operating environment constitute computer programs made up of computer-executable instructions. Describing the methods by reference to a flowchart enables one skilled in the art to develop such programs including such instructions to carry out the methods on suitable computers (the processor or processors of the computer executing the instructions from computer-readable media such as ROM, RAM, CD-ROM, hard disks, signals on network interfaces, etc.). The methods illustrated in FIGs. 3A and 3B are inclusive of acts that may be taken by an operating environment executing an exemplary embodiment of the invention.

FIG. 3A is a flowchart illustrating a method for failing over a network address from a first network interface on an application gateway device to a second network interface on an application gateway device according to an embodiment of the invention. In some embodiments, the network interfaces are Ethernet interfaces such as GE interface 211 described above.

The method begins when a system executing the method receives configuration data (block 305). In some embodiments, the configuration data includes the network addresses for applications running on the application gateway device, and may also include specifications of primary and secondary network interfaces that are to be assigned to the network address. In some embodiments, the network address is an IP network address.

At some point during the operation of the system, the system may detect the failure of a network interface (block 310). The failure may be detected either at startup time, in which case the secondary network interface may be used, or the failure may be detected after startup. In some embodiments of the invention, the failing network interface must be down for two seconds in order for a failure to be determined.

If the failure occurs after startup, the configuration data is analyzed to determine if the network address assigned to the first (failing) network interface can be failed over to the second network interface (block 315). Various embodiments of the invention may use various

factors in determining if the network address may be failed over from a first network interface to a second network interface. For example, one factor that may be analyzed is whether or not the network interfaces are connected to the same network. If not, the network address may not be failed over. Additionally, some embodiments of the invention analyze the configuration data to determine if the first and second network interfaces are on the same subnet. If not, the network address may not be failed over.

Additionally, some embodiments of the invention support VLANs (Virtual Local Area Network). In these embodiments, if the first network address and network interface are on a VLAN, the configuration data is analyzed to determine if second network interface can support the same VLAN. If not, the network address may not be failed over to the second network interface. In some embodiments executing the VTP protocol, a switch participating in the VLAN will inform the network interfaces which VLANs are acceptable. In alternative embodiments, the acceptable VLANS are configured.

Furthermore, in clustered environments, such as those described in FIG. 1B above, the system checks to see if the network address is in use by another application gateway device in the cluster. If so, the network address may not be failed over to the second network interface.

A further check performed by some embodiments of the invention is to determine if the second network interface can support an additional network address. In some embodiments, each network interface can support up to fifteen network addresses. If the second network interface is at the maximum, the network address may not be failed over.

Similarly, the system may check to determine if the second network interface can support an additional MAC address. If not, the network address may not be failed over.

After analyzing the configuration data as described above, the system will determine if the network address can be failed over from a failed first network interface to a second network interface (block 320). If so, the network address is moved to the second network interface (block 325) and applications using the first network interface continue to operate as if the failure did not occur (note that some data may need to be retransmitted, however this is typically handled by the network protocol layers and is typically transparent to the application). If not, the network address remains associated with the first network interface

and the application may no longer be able to send or receive data to and from the network.

FIG. 3B is a flowchart providing further details for block 325 above comprising a method for moving a network address from a first network interface to a second network interface according to an embodiment of the invention. The method begins by removing the network address from the failing first interface (block 340). In some embodiments, the network address is an IP address. In addition, a MAC address associated with the network address may also be removed (block 342).

Additionally, any static routes associated with the network address are removed from routing tables on the system (block 344).

In some embodiments of the invention, ARP (Address Resolution Protocol) entries associated with the first network address are removed from the system (block 346).

Finally, in some embodiments, any cached routes associated with the network address are flushed (i.e. removed) from the system (block 348). In some embodiments, cached routes associated with TCP, UDP and IP protocols are flushed.

The system then proceeds to prepare to associate the network address with the second network interface. The network address is assigned to the second network interface (block 350). In some embodiments, the MAC address that was associated with the network address on the first interface is moved to the second interface (block 352).

In some embodiments, the static routes that were removed at block 344 above are reinstalled on the system and associated with the second network interface (block 354).

In those embodiments supporting VLANS, if the first network interface was participating in a VLAN, then the VLAN logical interfaces are deleted from the first network interface and established on the second network interface if necessary.

Finally, in some embodiments of the invention, a gratuitous ARP packet is issued by the second network interface (block 356). The packet is gratuitous in that it is not issued in response to an ARP request. The gratuitous ARP is desirable, because it causes other network elements in the network such as switches and routers to update their respective ARP tables more quickly than they would through normal address resolution mechanisms that rely on timeouts.

It should be noted that the tasks performed above need not be performed in the order indicated in the flowchart. Additionally, various embodiments of the invention need not perform each and every task noted above.

Conclusion

Systems and methods for failing over a network address from a first network interface to a second network interface have been described. The embodiments of the invention provide advantages over previous systems. For example, by transferring the network address from one network interface to another, the failover may be transparent to the applications and hosts communicating with the applications, thus resulting in less disruption on the network.

While the embodiments of the invention have been described as operating in a storage router environment, the systems and methods may be applied to variety of application gateway devices, including switches, routers, personal computers, laptop computers, server computers etc. that have more than one network interface. This application is intended to cover any adaptations or variations of the present invention. The terminology used in this application is meant to include all of these environments. It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. Therefore, it is manifestly intended that this invention be limited only by the following claims and equivalents thereof.